UNIVERSITY OF SASKATCHEWAN
DEPARTMENT OF COMPUTATIONAL SCIENCE

CMPT 832.3 FINAL EXAM

3 hours Closed Book December 11, 1992

Answer ALL questions in an answer booklet, not on the exam paper. You may answer the questions in any order you like. Make sure that your name AND student number appear on EVERY examination booklet you hand in.

Use your time wisely and good luck. Before starting, make sure you have all 6 pages of the examination.

Please notice there is some choice! Don't do all questions!

A portion of the marks will be awarded for the style and clarity of the answers.

1. Write a Prolog predicate `intersection(X,Y,Z)` where `Z` is the intersection of `Y` and `Z`, subject to the constraint that `X`, `Y` and `Z` are sorted in ascending order, but may contain duplicates. In the event of duplicate elements `Z` contains as many copies of the element as appear in whichever of `X` or `Y` contain the most copies.

15

For example:

```
| ?- intersection([a,a,a,b,b,c],[a,a,b,b,b,b,d],X).

X = [a,a,b,b]
```

Write a *pure* Prolog program that uses no meta-predicates (i.e., `setof` or `bagof`). However, the use of arithmetic and relational operators is fine, as is the use of a properly documented red or green cut.

2. You have an iterative deepening Prolog, a breadth first Prolog, and a depth first Prolog meta interpreter.

   Which would you expect to be the best in each of the following cases? Justify each choice with a single sentence.

15

   (a) You have a very small program, guaranteed no loops, you want one answer.
   (b) You have a very small program, guaranteed no loops, you want all answers.
   (c) You have a substantial program, no loops, you want one answer.
   (d) You have a substantial program, possibly loops, you want one answer.
   (e) You have a substantial program, possibly loops, you want all answers.

3. Do **ONE** of the following **TWO** Questions.

   (a) THIS IS THE FIRST CHOICE, and has TWO parts.

      i. Suppose you have a Prolog meta interpreter that uses negated ancestor proofs. Consider the following program:

25

$a \vee b \vee c.$
$a \rightarrow x.$
$b \rightarrow x.$
$c \rightarrow x.$

$d \vee e.$
$d \rightarrow y.$
$e \rightarrow y.$

      How many proofs are there of $x \wedge y$? Why is this and can you generalize? Can you suggest a technique for reducing the number of proofs?

      ii. Consider the folowing story.

         Typically if I turn the ignition key the car starts. Typically if I turn the ignition key and the battery is dead, the car will not start. Typically if I leave the head lights on all night, the battery is dead.

      A Theorist representation of this problem might be

         default d1: tk $\rightarrow$ cs.
         default d2: tk $\wedge$ bd $\rightarrow \neg$ cs.
         default d3: lo $\rightarrow$ bd.
         fact lo $\wedge$ tk.

      A. What explanation(s) does Theorist give for *cs*?
      B. What explanation(s) does Theorist give for $\neg cs$?
      C. Suggest a constraint that will get rid of the answer *cs*.

(b) (THIS IS THE SECOND CHOICE)

When we are reasoning about time, a common default is that properties *persist*. That is, is something is true now, it is true at the next time point. For example, if I am alive now, I will be alive at the next time point. If a gun is loaded now, it is loaded at the next time point and was loaded at the time point preceding. If my hourse is red now, it is red at the next and previous time points. (25)

We might represent this in Theorist as follows:

default d1(X): alive(X) $\rightarrow$ alive(s(X)).
default d2(X): loaded(X) $\rightarrow$ loaded(s(X)).
default d3(X): alive(s(X)) $\rightarrow$ alive(X).
default d4(X): loaded(s(X)) $\rightarrow$ loaded(X).

where `s(X)` denotes the successor of time point `X`. If

We know the following facts about shooting:

fact shoot(X) and loaded(X) $\rightarrow \neg$ alive(s(X)).
fact shoot(X) and loaded(X) $\rightarrow \neg$loaded(s(X)).

This says that after shooting with a loaded gun, someone is no longer alive and the gun is no longer loaded.

Suppose given an initial time point 0, we know

fact loaded(0) and alive(0) and shoot(s(0)).

If you were to actually program this example, you would find a looping behaviour that is non-trivial to detect. So, construct your proof tree non deterministically.

i. Construct a Theorist proof tree for `not alive(s(s(0))) and not loaded(s(s(0)))`? Label the arcs with the names of the defaults. *You do not need to show the consistency proof or even justify it, but the combination of defaults you assume must be consistent!*

ii. Can you construct a Theorist proof of ~~not~~ `alive(s(s(0))) and not loaded(s(s(0)))`? If so, the conditions of the previous question apply. How did the gun become unloaded?

iii. Can you explain in your own words what the problem is with this example and suggest a solution?

4. Recall the model theory for propositional logic we discussed early in the year. A model M consisted of a set of worlds W and a truth function π from propositions and worlds into truth values. We wrote

$$\models^M_w A$$

if a formula A was true at a world w in M. We wrote

$$\models^M A$$

if a formula A was true at all worlds in M, and we wrote

$$\models A$$

when a formula is true at all worlds and all models.

25

(a) Give an example of a formula that is true in all worlds and all models. What is such a formula called?

(b) Give a model M and a formula A such that $\models^M A$, but not $\models A$.

(c) Let M be a model containing a set of worlds W and let A be a formula. Let us say a formula A is *possible* (which we write $P(A)$ iff there exists a world w in W such that

$$\models^M_w A.$$

A formula A is *necessary* (written $N(A)$) if for every w in W

$$\models^M_w A.$$

For each of the following, argue that they are true (keep it simple!) or give a counterexample:

i. If A is possible and $A \rightarrow B$ is possible, then B is possible.

ii. If A is necessary and $A \rightarrow B$ is possible, then B is possible.

iii. If A is possible and $A \rightarrow B$ is necessary, B is possible.

iv. If A is necessary and $A \rightarrow B$ is possible, it is necessary that B is possible.

5. Let us consider our "car starting problem", from the perspective of a Bayes net. Let cs, tk, lo represent the propositions "car start", "turn key", "battery dead" and "lights on".

$$\begin{tikzcd}
cs & \\
tk \arrow[u] & bd \arrow[ul] \\
& lo \arrow[u]
\end{tikzcd}$$

15

Suppose we know the following conditional probabilities:

p(cs|tk,bd) = 0.1
p(cs|tk,¬bd) = 0.99
p(cs|¬tk,bd) = 0.01
p(cs|¬tk,¬bd) = 0.02

p(bd|lo) = 0.95
p(bd|¬lo) = 0.3
p(lo) = 0.02
p(tk) = 0.95

What is

(a) $p(cs|tk, lo)$?

(b) $p(lo|tk, \neg cs)$?

6. Now consider using certainty factors to figure out whether the car starts. Suppose you have the following representation, where the number following the rule indicates the certainty:

10

```
turn-key => car-starts                0.9
weather-good => battery-good          0.5
battery-good => car-starts            0.9
not(battery-good) => car-starts      -0.9
lights-on => battery-good            -0.8
turn-key                              1.0
weather-good                          0.6
lights-on                             0.8
```